Accommodating Multiple Operating Systems and Memory Sizes on IA-32 Platforms

FIELD OF THE INVENTION

This invention relates generally to computer system programming, and more particularly to techniques for configuring computer system resources on IA-32 platforms.

BACKGROUND

Address Space, Cacheability, MTRR's, and the BIOS. Beginning with the introduction of the Pentium Pro, microprocessors in the IA-32 family provide a 36-bit address bus. This enables them to support up to 64GB of physical memory. All or portions of this physical memory may be cached in various ways in order to enhance performance. For example, one region of the address space may be designated as write-through, another may be designated as write-back, while still others may be designated as write-protected, write-combining or uncacheable. Assigning such cacheability characteristics to regions of the address space may be achieved by writing appropriate values into one or more pairs of memory type range registers ("MTRR's"). In most IA-32 computer systems, firmware known as the basic input-output system ("BIOS") begins executing prior to loading the operating system and configures certain resources within the computer including the MTRR's. Thus, it is generally a BIOS function to define caching behavior for regions within physical memory by programming the MTRR's.

Memory-Mapped IO. IA-32 processors permit applications to access input/output ("IO") ports in either of two ways: through a separate IO address space or through the physical memory address space. The latter approach is commonly

referred to as "memory-mapped IO." Accessing IO ports through the separate IO address space is achieved using a special set of IO instructions. Accessing IO ports through the physical memory address space, on the other hand, has the advantage that any of the processor's instructions that reference memory may be used to interact with IO devices.

Memory-mapped IO does present complications, however. One such complication arises in the context of cacheability: When controlling IO devices, it is usually important that IO operations be executed in precisely the order in which they are programmed to occur. Consequently, it is generally recommended that the portion of the physical address space to be used for memory-mapped IO should be designated as uncacheable. This is so because designating a memory-mapped IO region of the physical address space as uncacheable insures that reads from and writes to locations in the uncacheable region are carried out in program order.

Chipsets, the 4GB Boundary, and the Top of Lower Memory. Accesses to main memory by the CPU and other devices within a computer are generally handled by a memory controller chip--one of several chips commonly known as the "chipset." A chipset provides bus interface, data path, instruction caching and similar functions on the motherboard. The BIOS must configure the chipset at boot time with information about where main memory is located. If memory-mapped IO is in use, then the chipset must have information not only about where the actual main memory will be located, but also about where the memory-mapped IO region will be located within the physical memory address space.

Intel E7501 and similar chipsets for IA-32 platforms are designed to assume that certain devices will always be mapped into the peripheral component interconnect ("PCI") memory address range--that is, the address range beginning at 4GB and extending downward far enough to include a certain size of addresses.

Specifically, they assume that the advanced programmable interrupt controller ("APIC") addresses, the hub interface addresses, and any memory-mapped IO addresses will reside in this range. For purposes of this document, therefore, references made to "PCI memory," the "PCI memory address range," or the "PCI range" shall mean the range of addresses beginning at 4GB (actually 4096MB) and extending downward far enough to include the size of the APIC addresses, the hub interfaces addresses, and the memory-mapped IO addresses.

Configuration of these chipsets requires among other things that the BIOS write appropriate values into the top of lower memory ("TOLM") register and into the DRAM row boundary 7 ("DRB7") register. The TOLM register is designed to contain the maximum address below 4GB that should be treated as main memory. The DRB7 register is designed to contain the maximum address in the machine that should be treated as main memory. Thus, for machines having less physical memory than 4GB minus the minimum size required for the PCI memory address range, the TOLM and DRB7 registers will contain the same value. But for machines having more memory than that, the physical memory must be split because the PCI memory address range may not be moved. In such machines, there will be one region of physical memory located below the PCI range, and another region of physical memory located above the PCI range. The TOLM register will indicate the highest address within the first range. The DRB7 register will indicate the highest address within the second range.

Prior Art Recommendation for Setting TOLM. In reference to setting the value to be contained by the TOLM register, the E7501 chipset data sheet contains the following recommendation: "Configuration software should set this value to either the maximum amount of memory in the system or to the minimum address allocated for PCI memory, whichever is smaller." In other words, the prior art

recommendation for setting TOLM is to choose a value that will maximize the amount of physical memory located below the PCI range, regardless of the amount of physical memory available on the machine.

SUMMARY OF THE INVENTION

In one aspect, a method of configuring IA-32 computer resources according to a preferred embodiment of the invention includes determining the amount of physical memory available in the computer and the minimum total size required for memory-mapped IO. If the amount of available physical memory is less than or equal to 4GB, then the minimum required memory-mapped IO size is rounded up to the next multiple of 128M. Otherwise, the minimum required memory-mapped IO size is rounded up to the next multiple of x, where x is determined responsive to the number of DIMM socket pairs available in the computer. The top of lower memory is then set equal to 4GB minus the rounded memory-mapped IO size.

In one embodiment, the function for determining x may be as follows: when the number of DIMM socket pairs available is 2, x equals 256MB; when the number of DIMM socket pairs available is 3, x equals 512MB; when the number of DIMM socket pairs available is 4, x equals 1024MB; when the number of DIMM socket pairs available is greater than 4, x equals 2GB; otherwise, the value of x is undefined.

Configuring the resources of a computer in accordance with the invention yields the advantage that multiple operating systems and memory sizes are accommodated automatically on IA-32 platforms. For operating systems that can address only up to 4GB of the address space, the inventive method chooses an acceptable value for TOLM because the granularity used to choose the value is reasonably small--128MB. For operating systems that can address more than 4GB of the address space, the method chooses an exceptionally good value for TOLM in

the following sense: Regardless of the amount of physical memory available on the machine, the value chosen for TOLM will enable the BIOS to define caching characteristics for all of the memory using a maximum of 6 of the available 8 pairs of MTRR's. Moreover, the BIOS is enabled to do so without the use of overlapping techniques that may impose unwanted caching characteristics on the PCI memory address range. This is important because some operating systems--Linux, for example--require the use of 2 MTRR's to define the caching characteristics of regions within the PCI memory address range that are used for high-performance graphics operations.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a flow diagram illustrating a method of configuring computer resources according to a preferred embodiment of the invention.

Fig. 2 is a flow diagram illustrating a preferred method of determining x in the diagram of Fig. 1.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 illustrates a method 100 of configuring IA-32 computer resources according to a preferred embodiment of the invention. In one embodiment, the method was implemented in the BIOS. In alternative embodiments, the method may be implemented in other firmware or in software or hardware. In step 102, the method determines the amount of physical memory available in the computer. In step 104, the method determines the minimum size needed to contain the memory-mapped IO area. In step 106, a test determines whether the amount of physical memory available is less than or equal to 4GB. If so, then the minimum memory-mapped IO

size is rounded up to the next multiple of 128MB in step 108, and execution continues with step 116. But if not, then execution continues with step 110.

In step 110, the method determines the number of DIMM socket pairs available in the computer. In step 112, the method determines a value x as a function of the number of DIMM socket pairs available. In step 114, the minimum memory-mapped IO size is rounded up to the next multiple of x. In step 116, the value of TOLM is set equal to 4GB minus the rounded up memory-mapped IO size.

Fig. 2 illustrates a preferred method 200 for determining the value of x in step 112. In steps 202 and 204, if the number of DIMM socket pairs available is 2, then x is set to 256MB and the method returns. Else, in steps 206 and 208, if the number of DIMM socket pairs available is 3, then x is set to 512MB and the method returns. Else, in steps 210 and 212, if the number of DIMM socket pairs available is 4, then x is set to 1024MB and the method returns. Else, in steps 214 and 216, if the number of DIMM socket pairs available is greater than 4, then x is set to 2GB and the method returns. Any other numbers of DIMM socket pairs available (such as 1) would be an error condition, as indicated at step 218.

After determining the value of TOLM in accordance with the invention, MTRR pairs may be programmed to define caching characteristics for physical memory. Preferably, the MTRR's programmed by the BIOS should not dictate caching characteristics for the memory-mapped IO region. This is so because some operating systems, such as Linux, prefer to do so themselves after the BIOS has configured the rest of the system. In addition, care should be taken so that the BIOS uses at most 6 pairs of MTRR's when defining the caching characteristics for physical memory. This is so because some operating systems, such as Linux, that assign caching characteristics to the memory-mapped IO region, require 2 MTRR pairs to

do so. In this regard, note that an MTRR pair is only capable of defining a memory region having a size that is equal to an even power of 2.

Consideration of a few examples will illustrate how the method of the invention successfully achieves the above objectives in a variety of worst-case circumstances. In one case, assume that total physical memory is determined to be 4GB (actually 4096MB), and that the minimum memory-mapped IO size is less than 128MB. Memory-mapped IO size is rounded up to 128MB in step 108. In step 116, TOLM is set to 3968MB (4096MB minus 128MB). This leaves 128MB of physical memory above the PCI memory address range. A total of 6 MTRR pairs would be required to define caching characteristics for physical memory. Five MTRR pairs would define caching characteristics below the memory-mapped IO region as follows: 3968MB = 2048MB + 1024MB + 512MB + 256MB + 128MB. One MTRR pair would define characteristics for the 128MB above the PCI memory address range as follows: 128MB = 128MB.

In another case, assume that total physical memory is determined to be 6GB, that the number of DIMM socket pairs available is 2, and that the minimum memory-mapped IO size is less than 256MB. Memory-mapped IO size is rounded up to 256MB in step 114. In step 116, TOLM is set to 3840MB (4096MB minus 256MB). This leaves 2560MB of physical memory above the PCI memory address range. A total of 6 MTRR pairs would be required to define caching characteristics for physical memory. Four MTRR pairs would define caching characteristics below the memory-mapped IO region as follows: 3840MB = 2048MB + 1024MB + 512MB + 256MB. Two MTRR pairs would define characteristics for the 2560MB above the PCI memory address range as follows: 2560MB = 2048MB + 512MB.

In another case, assume that total physical memory is determined to be 7GB, that the number of DIMM socket pairs available is 3, and that the minimum memory-

mapped IO size is less than 512MB. Memory-mapped IO size is rounded up to 512MB in step 114. In step 116, TOLM is set to 3584MB (4096MB minus 512MB). This leaves 3584MB of physical memory above the PCI memory address range. A total of 6 MTRR pairs would be required to define caching characteristics for physical memory. Three MTRR pairs would define caching characteristics below the memory-mapped IO region as follows: 3584MB = 2048MB + 1024MB + 512MB. Three MTRR pairs would define characteristics for the 3584MB above the PCI memory address range as follows: 3584MB = 2048MB + 1024MB + 512MB.

In yet another case, assume that total physical memory is determined to be 14GB, that the number of DIMM socket pairs available is 4, and that the minimum memory-mapped IO size is less than 1024MB. Memory-mapped IO size is rounded up to 1024MB in step 114. In step 116, TOLM is set to 3072MB (4096MB minus 1024MB). This leaves 11264MB of physical memory above the PCI memory address range. A total of 5 MTRR pairs would be required to define caching characteristics for physical memory. Two MTRR pairs would define caching characteristics below the memory-mapped IO region as follows: 3072MB = 2048MB + 1024MB. Three MTRR pairs would define characteristics for the 11264MB above the PCI memory address range as follows: 11264MB = 8192MB + 2048MB +1024MB.

While the invention has been described in detail in relation to preferred embodiments thereof, the described embodiments have been presented by way of example and not by way of limitation. It will be understood by those skilled in the art that various changes may be made in the form and details of the described embodiments, resulting in equivalent embodiments that remain within the scope of the appended claims.